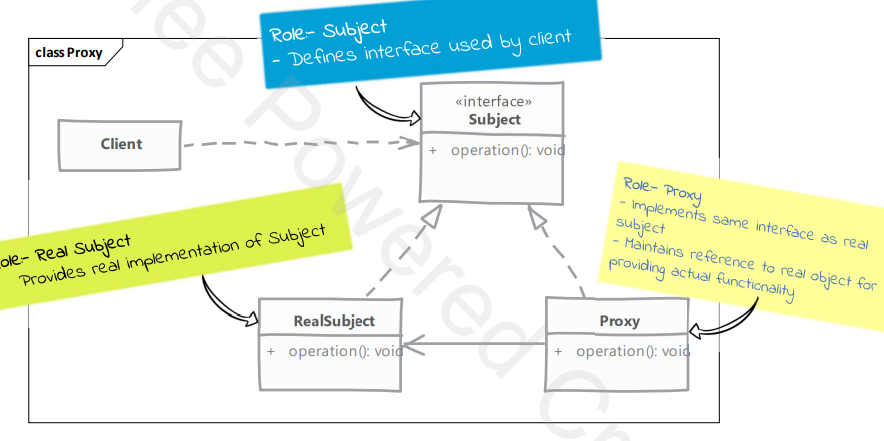
Proxy Pattern

Structural Design Pattern

**What is Proxy:**

1. When we need to provide a placeholder or surrogate to another object.
2. Proxy acts on behalf of the object and is used for lots of reasons some of the main reasons are:
   1. Protection Proxy - Control access to original object’s operations
   2. Remote Proxy - Provides a local representation of a remote object.
   3. Virtual Proxy - Delays construction of original object until absolutely necessary.
3. Client is unaware of the existence of proxy. Proxy performs its work transparently.

**UML:**



**Implementation Steps of Static Proxy:**

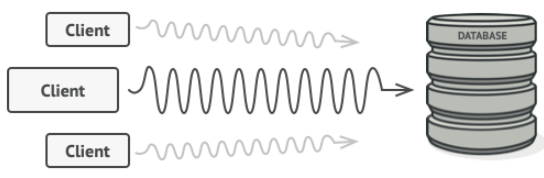
1. Implement Proxy:
   1. Proxy must implement same interface as the real subject
   2. Either create an actual object later when required or ask for one in the constructor.
   3. In method implementations of proxy we implement proxy’s functionality before delegating to a real object..
2. How to provide clients with proxies instances is decided by the application. We can provide a factory or compose client code with proxies instance.

**Implementation steps of Dynamic Proxy:**

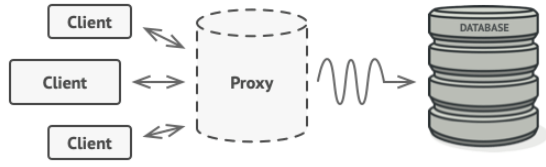
1. Allows us to create proxies at runtime.
2. Implementing java.lang.reflect.InvocationHandler
3. Invocation handler implements invoke method which is called to handle every method invocation on proxy

**Intent:** lets you provide a substitute or placeholder for another object. A proxy controls access to the original object, allowing you to perform something either before or after the request gets through to the original object.

**Problem:** You have a massive object that consumes a vast amount of system resources. You need it from time to time, but not always. You could implement lazy initialization: create this object only when it’s actually needed. All of the object’s clients would need to execute some deferred initialization code. Unfortunately, this would probably cause a lot of code duplication.



**Solution:** The Proxy pattern suggests that you create a new proxy class with the same interface as an original service object. Then you update your app so that it passes the proxy object to all of the original object’s clients. Upon receiving a request from a client, the proxy creates a real service object and delegates all the work to it. Proxy lets you execute something either before or after the primary logic of the class, without changing the class.



**Real-World Analogy:** A credit card is a proxy for a bank account, which is a proxy for a bundle of cash. Both implement the same interface. You don’t need to carry loads of cash around, only a card. Credit cards can be used for payments just the same as cash.

